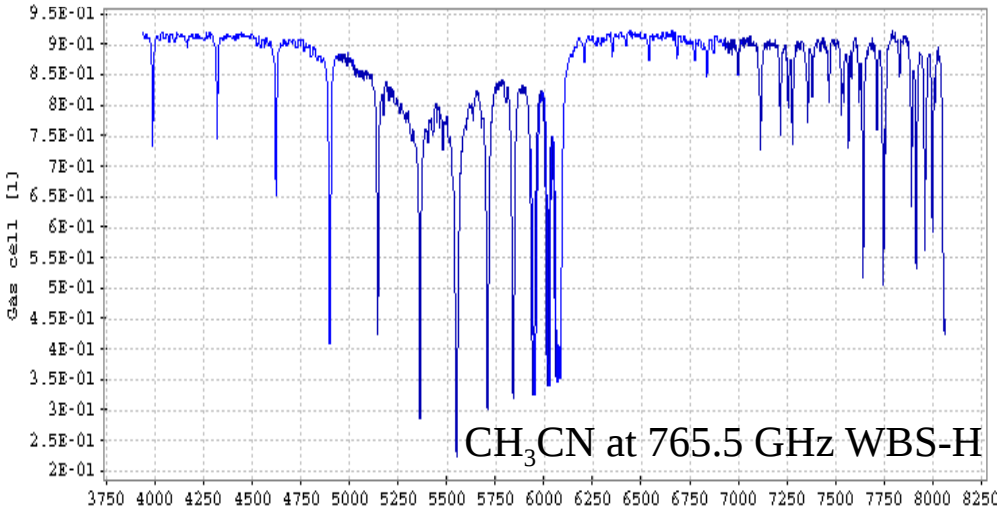
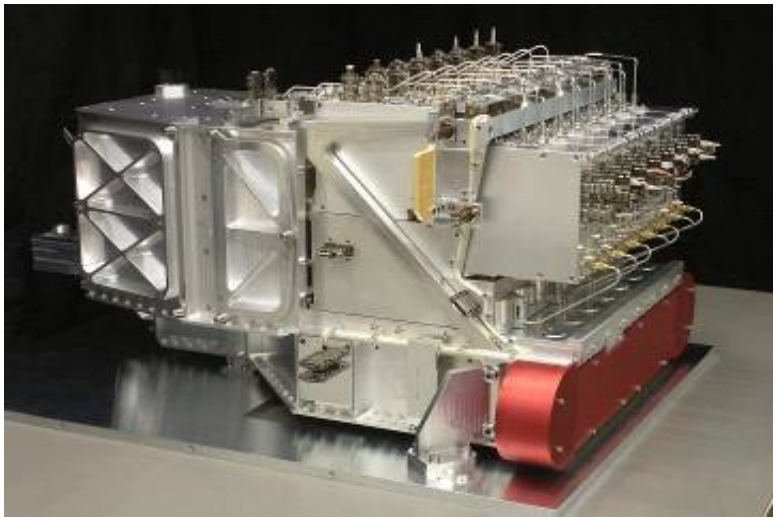
Exporting HIFI spectra to CLASS

- HiClass exports HIFI spectra from HCSS to CLASS.
- Used a lot recently since we collected interesting data
    - => user feedback.
- HIClass creates a CLASS readable FITS file.
- HiClass is offered as an application for HIFI SpectrumDatasets.
- Some limitations:
  - CLASS expects regular frequency grid
  - Data need to be resampled, stitched and space craft velocity corrected before export.

Standing Wave Removal

Adwin Boogert, NHSC/IPAC, Pasadena, CA, USA



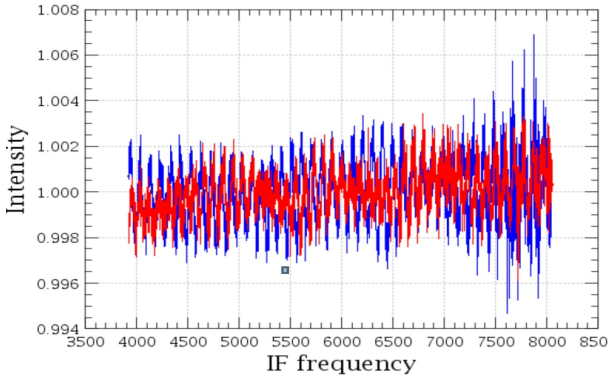


Netherlands Institute for Space Research

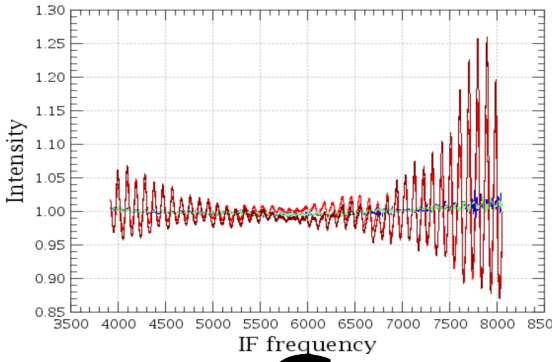
Level 1 → 2 Processing
Standing Wave Removal: the Problem

Standing waves from optical and electronic components seen (as expected) in HIFI spectra. Pipeline removes those by using appropriate chopping against sky or load, nodding, or frequency switch. Standing wave residuals might still be seen in Level 2 data. Wave-type is HIFI-band dependent:

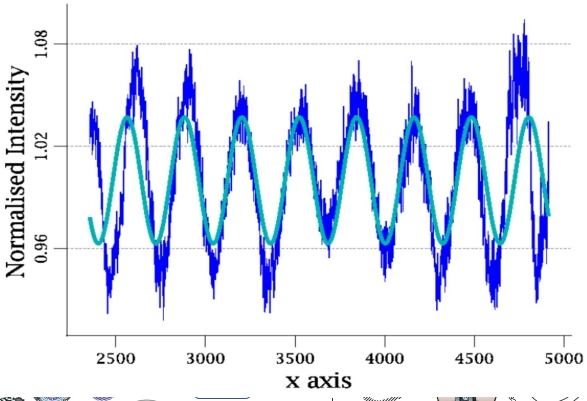
- Beamsplitter bands 1, 2, and 5 show sine waves



- Diplexer bands 3 and 4 show sine waves with amplitude increasing to IF band edges



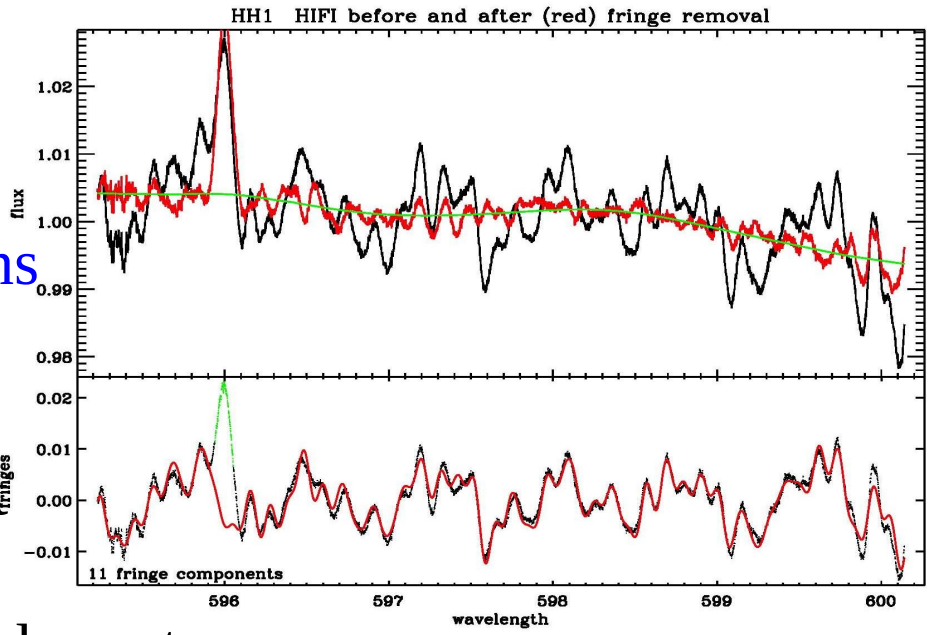
- HEB bands 6 and 7 waves are not sine waves. Requires special treatment, not discussed here.



Netherlands Institute for Space Research

Level 1 → 2 Processing
Standing Wave Removal: Principles

Steps to remove ***residual*** standing waves (SW), ***with as little user-interaction as possible:***



- Separate SW from:
  - other baseline fluctuations
  - emission/absorption lines

- Fit N sine waves with different periods, amplitudes, phases to baseline-subtracted, line-masked spectrum.

- Subtract SW fit from original input data. Caveat: should be division, but often not possible as 'continuum' level uncertain. Not big effect as SW amplitude <few%

Netherlands Institute for Space Research

Level 1 → 2 Processing
Standing Wave Removal: Principles

- Baseline and mask determined using iterative filtering, smoothing, sigma-clipping algorithm. Needs to know 'typical' longest standing wave period, to separate waves from other baseline structure.
- N sine waves A*sin(x)+B*cos(x)=C*sin(x+phase) fitted to baseline-subtracted, line-masked spectra iteratively:

A) Determine period, amplitude, phase of wave 1 at minimum chi^2

B) Determine period wave 2 at minimum chi^2 in sine-wave subtracted data

C) Solve for amplitude and phase ***combined waves 1+2*** using 'LU' matrix decomposition in original input data

D) Subtract ***waves 1+2*** from original input data

E) Repeat steps B, C, D for waves 1+2+3, ..., 1+2+3+...+N

Level 1 → 2 Processing
Standing Wave Removal: in HIPE

•FitHifiFringe: HIFI standing wave removal tool. Available in HIPE 1.2 and 2.0, but still being improved based on ongoing PV experience. Makes use of core tasks:

- FitFringe: fits sine waves to SpectralSegments (basically, any wavelength [in micron!], flux spectrum). Not instrument specific, i.e. could be adapted for use with PACS and SPIRE spectra.

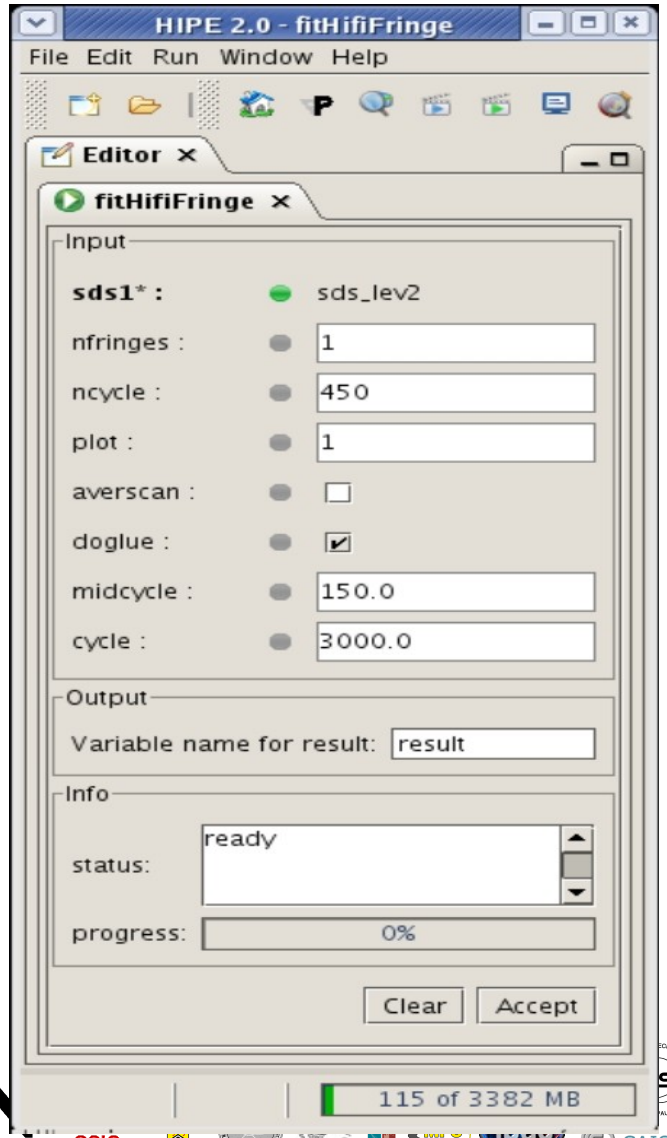
- SmoothBaseline: used by FitFringe, but can be used in standalone mode. Returns baseline AND line mask. NOT a polynomial baseline fit, but rather a smoothed, clipped version of the spectrum. Use with care!

Level 1 → 2 Processing
Standing Wave Removal: in HIPE

Most relevant FitHifiFringe input:

- WBS or HRS SpectrumDataset
- nfringes: number of sine waves to fit
- Plot: 0=not, 1=most important plots
- Averscan: determine SW from average of all scans, and subtract that from all. Might be useful if waves are weak.
- Doglue: determine SW on combined WBS sub-bands
- Midcycle: typical SW expected in data [MHz] Longer period structures are assumed to be baseline or sky features.
- Cycle: longest period SW to search for [MHz]

Output: sine wave(s)-subtracted SpectrumDataset and list of sine wave parameters fitted

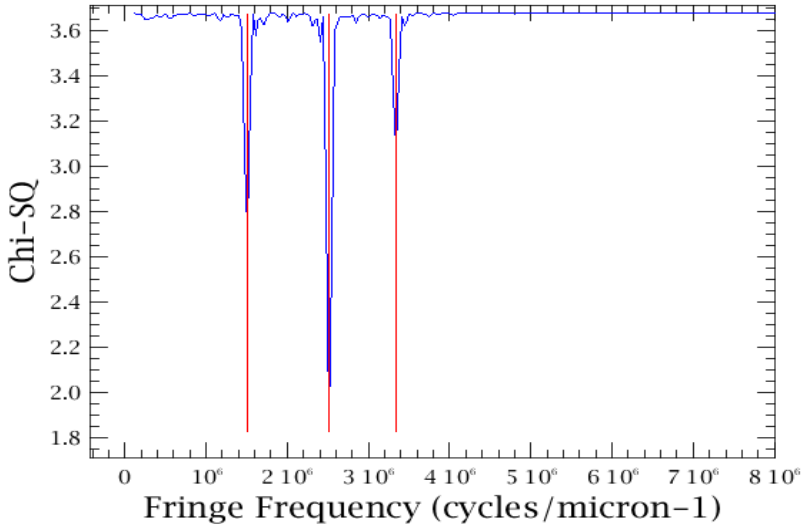


Netherlands Institute for Space Research

Level 1 → 2 Processing
Standing Wave Removal: in HIPE

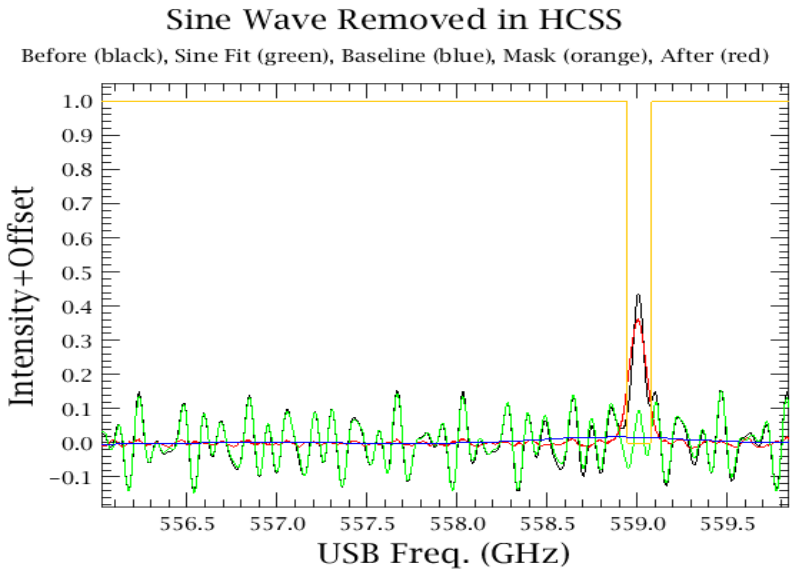
FitHifiFringe shows fitted periods, amplitudes, phases in HIPE console and by default produces 2 plots:

- Chi^2 as function of period. Minima found are indicated with red vertical lines



(HIPE 3.0 will have Period in MHz!)

- Result plot: input data, sine wave fit, baseline, mask, sine-wave subtracted spectrum



Netherlands Institute for Space Research

Level 1 → 2 Processing
Standing Wave Removal: Future

- FitHifiFringe will keep evolving as more PV data becomes avaivable and the 'problem' becomes better constrained
- HIPE 3.0: manual masking possible and ObsContext in/output
- Suggestions welcome
- Removing band 6 and 7 waves a different problem, which cannot be solved by fitting sine waves. Alternative methods are being worked on.

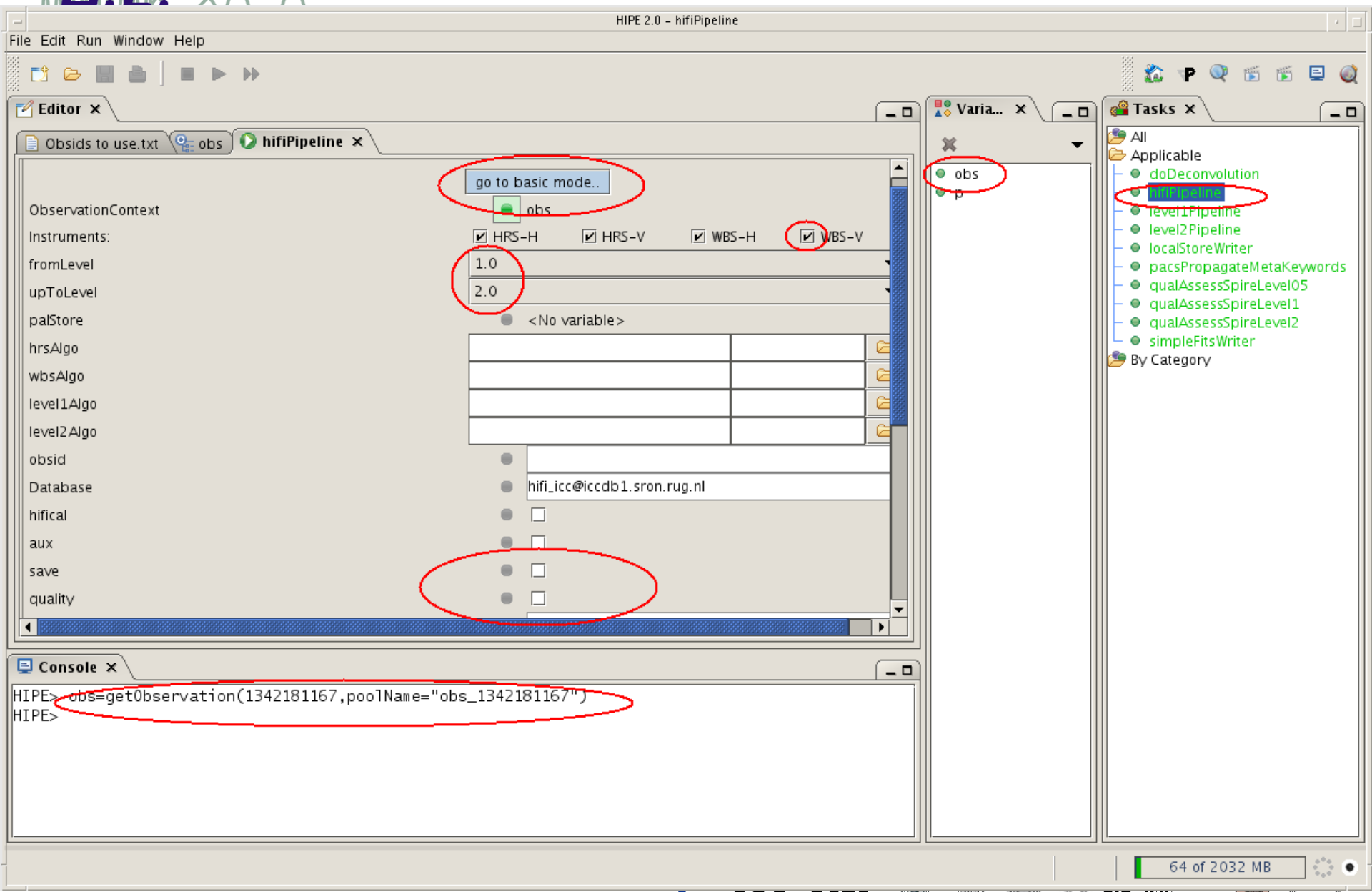
Adjusting the HIFI Pipeline

- The most appropriate level to make changes in the HIFI pipeline is processing from Level 1 to Level 2. At this level is may be necessary to:
  - Resample onto user specified grid.
  - Stitch subbands together
  - Correct for standing waves
  - Remake spectral cubes
  - Run a spectral survey through the deconvolution tool
  - Export to CLASS
  - Etc...
- The pipeline allows to introduce these changes my altering the Level2PipelineAlgo.py
  - The file is found in

    <hipe directory>/scripts/hifi/pipeline/generic/Level2PipelineAlgo.py

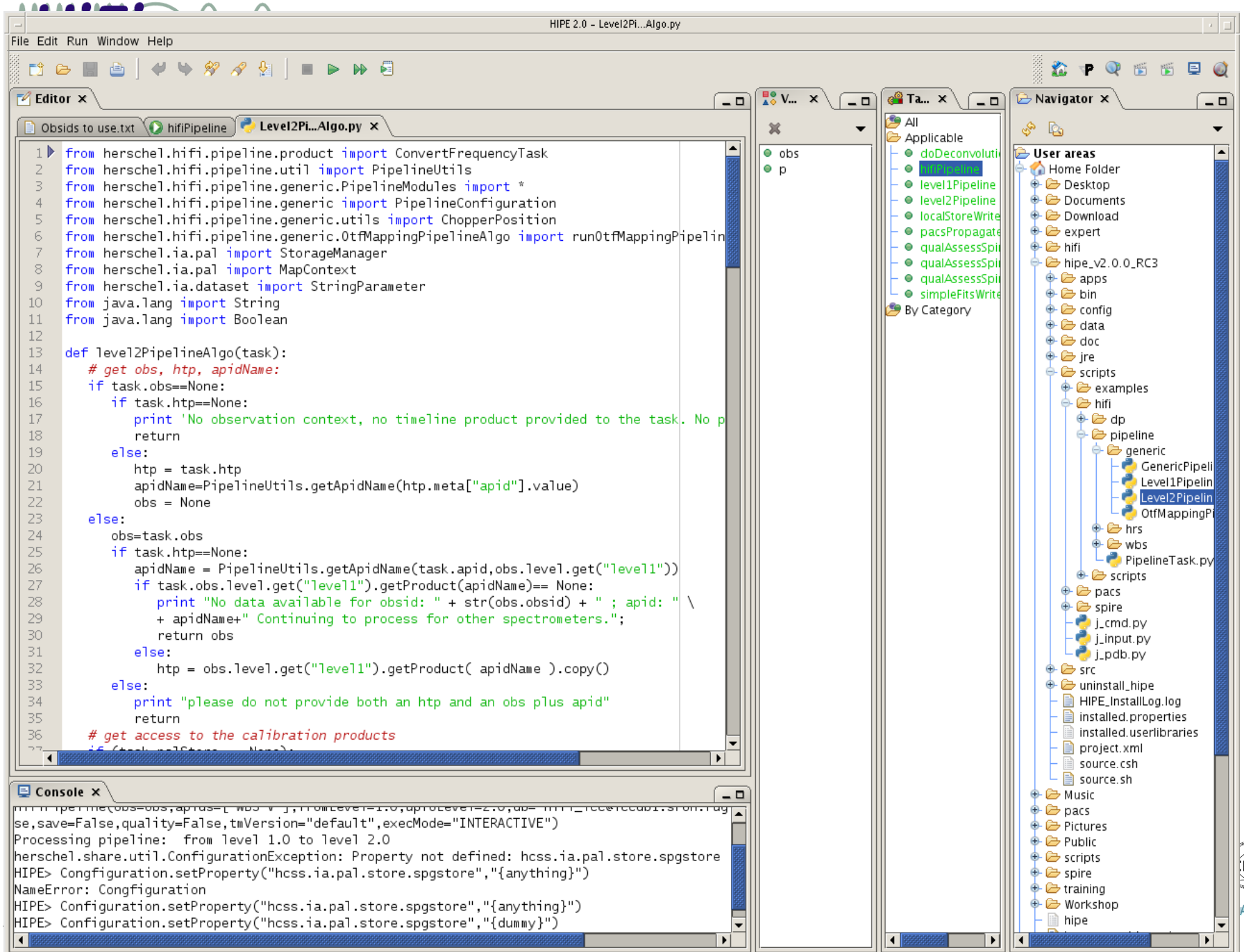
How to setup HifiPipelineTask

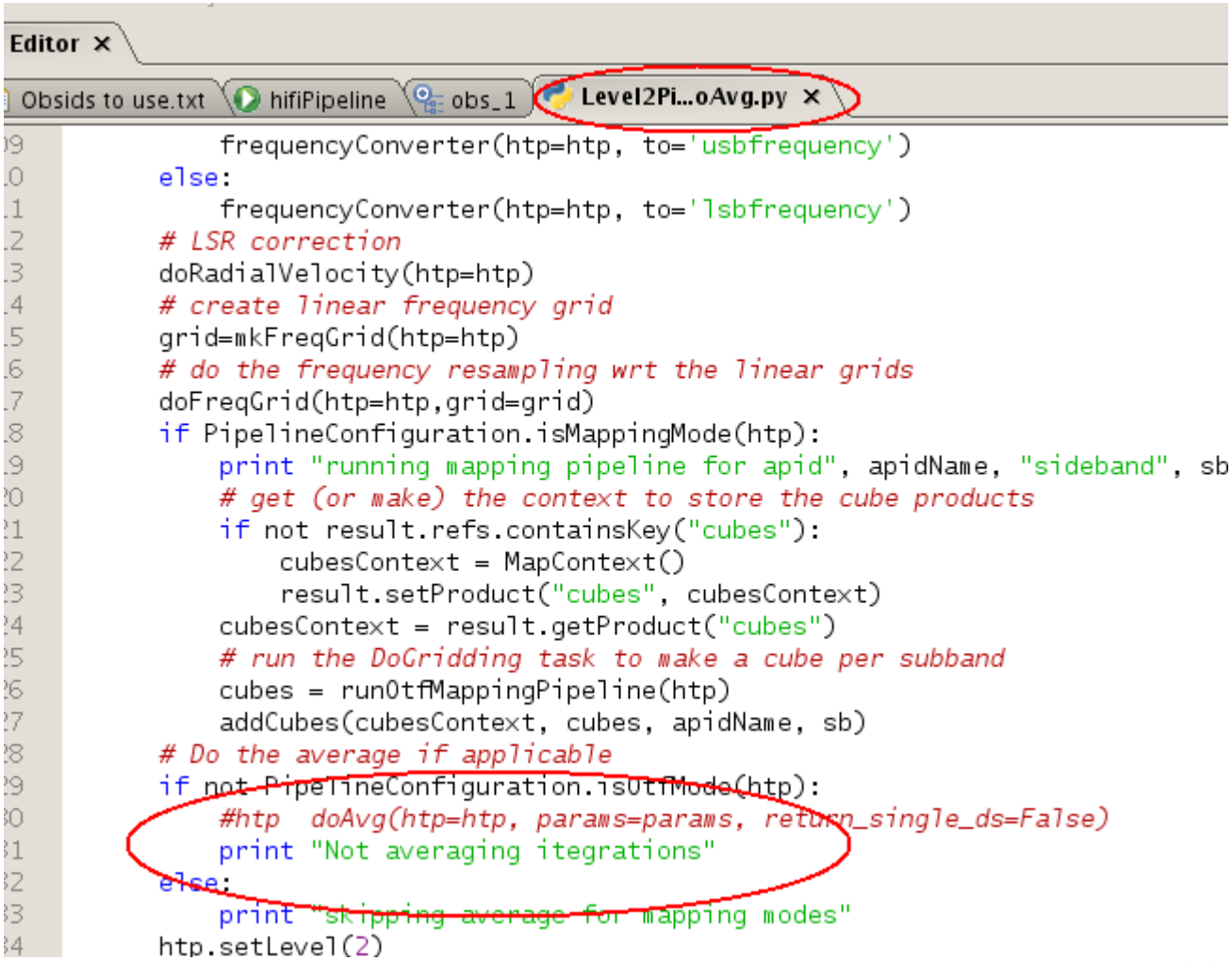
- HifiPipelineTask is the jython task which controls the HIFI pipeline.
  - This task is activated by an observation context present in the Hipe session.
  - For the Hipe 2.0 RC3 version use the following:
    - Change to Expert Mode
    - Turn off palStore option
    - Turn off Quality plugin.
    - Type in: Configuration.setProoperty(“hcss.ia.pal.store.spgstore”,”{dummy}”)
  - Select run levels (for this example, I set fromLevel to 1.0 and upTo level to 2.0
  - Press “Accept” to rerun the pipeline.

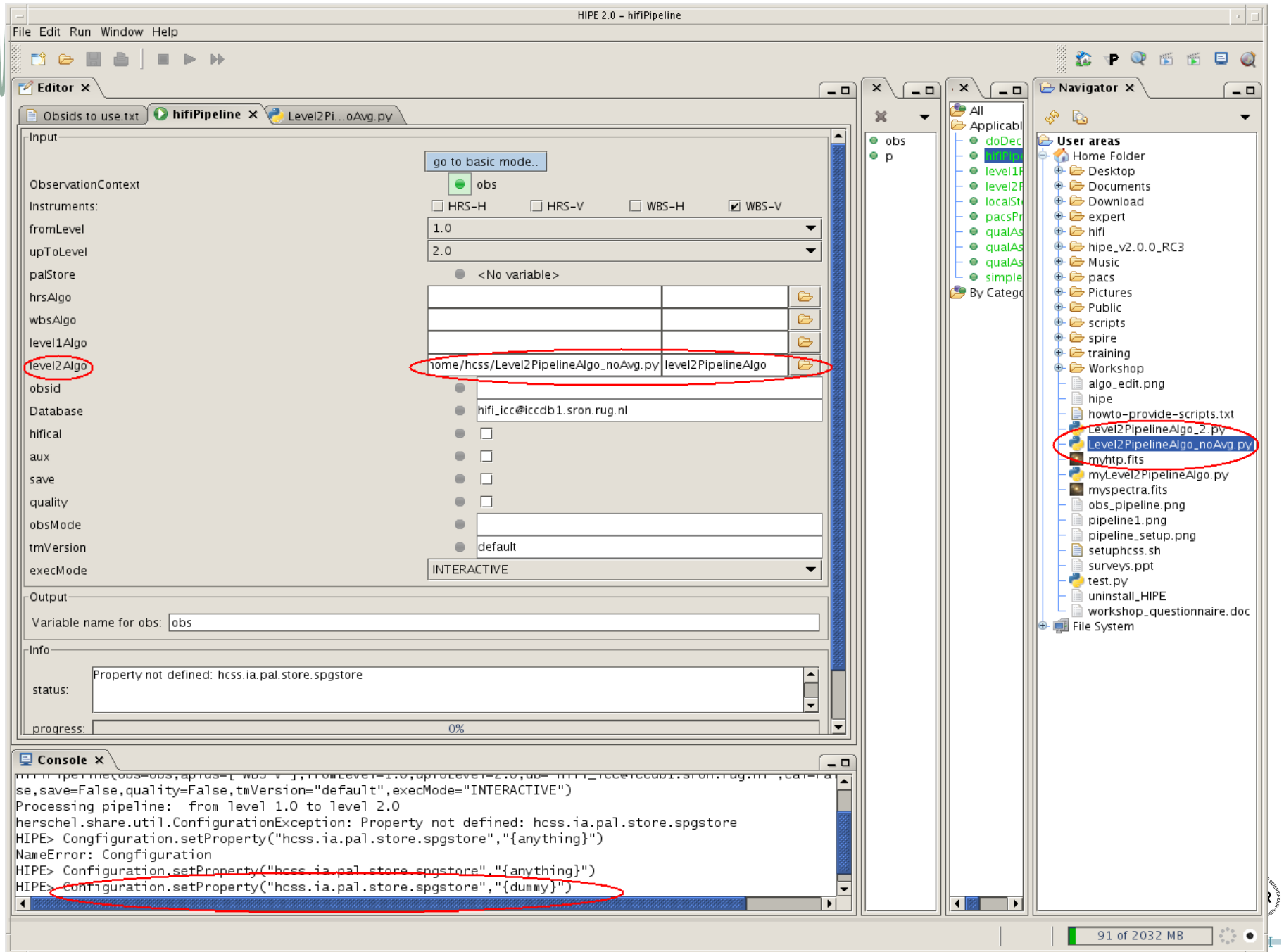


Altering and applying an updated Algorithm

- Copy the Level2PipelineAlgo.py to your own area and rename the file.
- Edit the algorithm file.
  - In my example I just comment out the doAvg task. And add a line for the pipeline which reflects that I've done this.
- In the leve2Algo input window for the task dialog, add the new file name for the algorithm you have just edited.
- Set fromLevel to 1.0 and upToLevel to 2.0 in the dialog box.
- Run the pipeline by pressing “Accept” in the dialog box

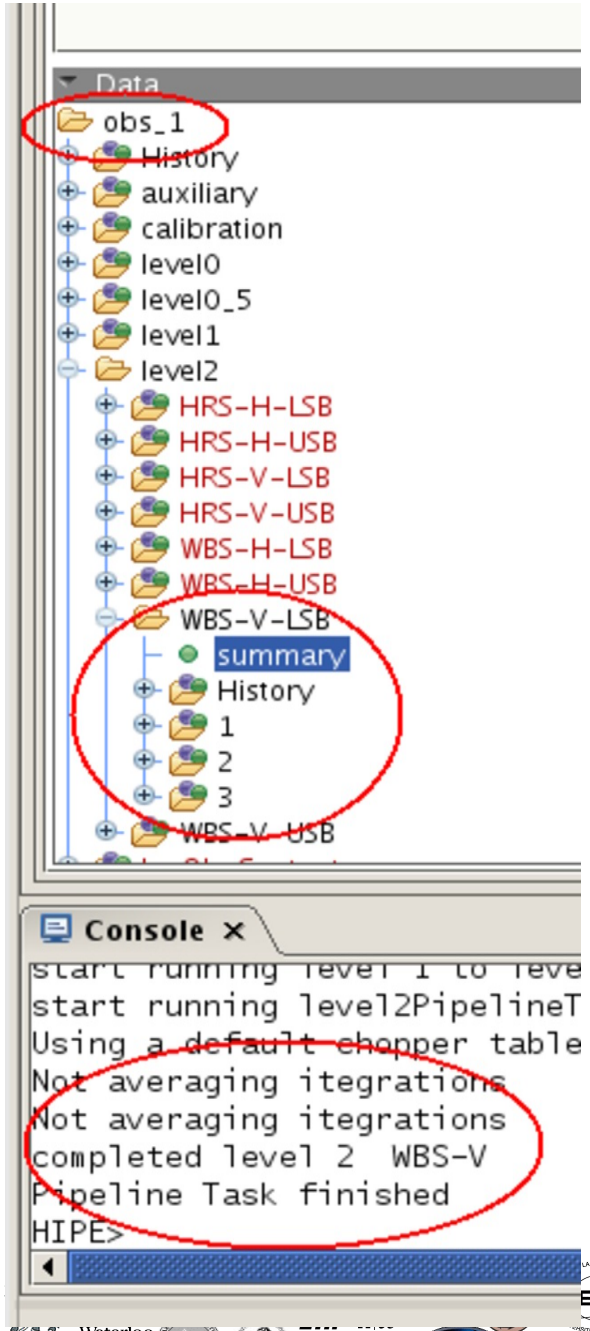

HIPE 2.0 - Level2Pi...Algo.py
File Edit Run Window Help
Editor
Obsids to use.txt
hifiPipeline
Level2Pi...Algo.py
from herschel.hifi.pipeline.product import ConvertFrequencyTask
from herschel.hifi.pipeline.util import PipelineUtils
from herschel.hifi.pipeline.generic.PipelineModules import *
from herschel.hifi.pipeline.generic import PipelineConfiguration
from herschel.hifi.pipeline.generic.utils import ChopperPosition
from herschel.hifi.pipeline.generic.OtfMappingPipelineAlgo import runOtfMappingPipelin
from herschel.ia.pal import StorageManager
from herschel.ia.pal import MapContext
from herschel.ia.dataset import StringParameter
from java.lang import String
from java.lang import Boolean
def level2PipelineAlgo(task):
# get obs, htp, apidName:
if task.obs==None:
if task.htp==None:
print 'No observation context, no timeline product provided to the task. No p
return
else:
htp = task.htp
apidName=PipelineUtils.getApidName(htp.meta["apid"].value)
obs = None
else:
obs=task.obs
if task.htp==None:
apidName = PipelineUtils.getApidName(task.apid,obs.level.get("level1"))
if task.obs.level.get("level1").getProduct(apidName)== None:
print "No data available for obsid: " + str(obs.obsid) + " ; apid: " \
+ apidName+" Continuing to process for other spectrometers.";
return obs
else:
htp = obs.level.get("level1").getProduct(apidName).copy()
else:
print "please do not provide both an htp and an obs plus apid"
return
get access to the calibration products
Console
se,save=False,quality=False,tmVersion="default",execMode="INTERACTIVE")
Processing pipeline: from level 1.0 to level 2.0
herschel.share.util.ConfigurationException: Property not defined: hcss.ia.pal.store.spgstore
HIPE> Congfiguration.setProperty("hcss.ia.pal.store.spgstore","{anything}")
NameError: Congfiguration
HIPE> Configuration.setProperty("hcss.ia.pal.store.spgstore","{anything}")
HIPE> Configuration.setProperty("hcss.ia.pal.store.spgstore","{dummy}")
V...
obs
p
Ta...
All
Applicable
doDeconvoluti
hifiPipeline
level1Pipeline
level2Pipeline
localStoreWrite
pacsPropagate
qualAssessSpi
qualAssessSpi
qualAssessSpi
simpleFitsWrite
By Category
Navigator
User areas
Home Folder
Desktop
Documents
Download
expert
hifi
hipe_v2.0.0_RC3
apps
bin
config
data
doc
jre
scripts
examples
hifi
dp
pipeline
generic
GenericPipeli
Level1Pipelin
Level2Pipelin
OtfMappingPi
hrs
wbs
PipelineTask.py
scripts
pacs
spire
j_cmd.py
j_input.py
j_pdb.py
src
uninstall_hipe
HIPE_InstallLog.log
installed.properties
installed.userlibraries
project.xml
source.csh
source.sh
Music
pacs
Pictures
Public
scripts
spire
training
Workshop
hipe




HIPE 2.0 - hifiPipeline
File Edit Run Window Help
Editor
Obsids to use.txt
hifiPipeline
Level2Pi...oAvg.py
Input
go to basic mode..
ObservationContext
obs
Instruments:
HRS-H
HRS-V
WBS-H
WBS-V
fromLevel
1.0
upToLevel
2.0
palStore
<No variable>
hrsAlgo
wbsAlgo
level1Algo
level2Algo
ome/hcss/Level2PipelineAlgo_noAvg.py
level2PipelineAlgo
obsid
Database
hifi_icc@iccdb1.sron.rug.nl
hifical
aux
save
quality
obsMode
tmVersion
default
execMode
INTERACTIVE
Output
Variable name for obs:
obs
Info
status:
Property not defined: hcss.ia.pal.store.spgstore
progress:
0%
Console
se,save=False,quality=False,tmVersion="default",execMode="INTERACTIVE")
Processing pipeline: from level 1.0 to level 2.0
herschel.share.util.ConfigurationException: Property not defined: hcss.ia.pal.store.spgstore
HIPE> Congfiguration.setProperty("hcss.ia.pal.store.spgstore","{anything}")
NameError: Congfiguration
HIPE> Configuration.setProperty("hcss.ia.pal.store.spgstore","{anything}")
HIPE> Configuration.setProperty("hcss.ia.pal.store.spgstore","{dummy}")
obs
p
All
By Categ
Navigator
User areas
Home Folder
Desktop
Documents
Download
expert
hifi
hipe_v2.0.0_RC3
Music
pacs
Pictures
Public
scripts
spire
training
Workshop
algo_edit.png
hipe
howto-provide-scripts.txt
Level2PipelineAlgo_2.py
Level2PipelineAlgo_noAvg.py
myhtp.fits
myLevel2PipelineAlgo.py
myspectra.fits
obs_pipeline.png
pipeline1.png
pipeline_setup.png
setuphcss.sh
surveys.ppt
test.py
uninstall_HIPE
workshop_questionnaire.doc
File System
91 of 2032 MB

Final spectrum

- Running the pipeline again using this algorithm, shows that final spectra are not averaged.
-



Netherlands Institute for Space Research